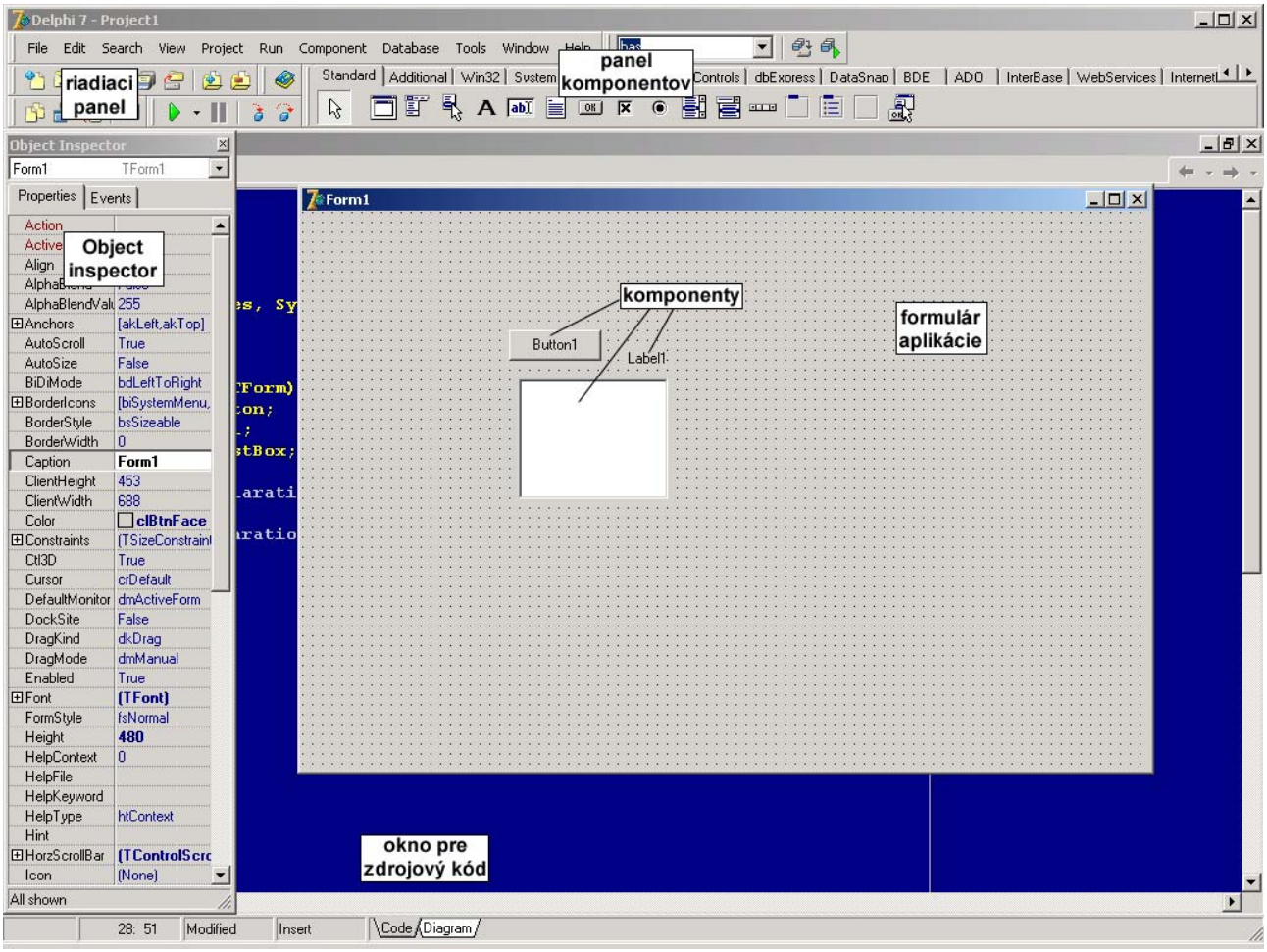
Borland Delphi



Borland Delphi je vývojové prostredie primárne určené na tvorbu aplikácií pod Windows. Je založené na mutácii jazyka Object Pascal.
Tvorba aplikácie v Delphi je oproti ekvivalentom používaným v prostredí MS DOS (Pascal) na prvý pohľad veľmi odlišná, pretože prostredie aplikácie je štandardne tvorené formulárom, na ktorý sa umiestňujú ostatné prvky – komponenty (relatívne uzavretý prvok, ktorý má svoje vlastnosti a je schopný reagovať na rozličné udalosti – kliknutie, prechod myšou, stlačenie tlačidla a pod.).
Filozofia programovania je založená na spracúvaní udalostí, ktoré sa v aplikácii v rámci jednotlivých komponentov odohrávajú – z toho pramení aj označenie *udalosťami riadené programovanie*.

Po spustení aplikácie sa dostaneme do prostredia pozostávajúceho z hlavných častí:

- *riadiaci panel:* slúži na manipuláciu so súbormi Delphi (otvorenie, uloženie, spustenie a zastavenie behu aplikácie)
- *formulár aplikácie*: predstavuje to, čo vo Windows poznáme pod pojmom okno. Samotný formulár je vlastne okno (niekedy celá aplikácia), ktorá môže (ale nemusí) obsahovať ďalšie prvky,
- *zoznam komponentov*: obsahuje časť komponentov, ktoré možno používať pri tvorbe aplikácie. Na formulár sa prenášajú kliknutím na komponent a kliknutím na formulára alebo dvojklikom na komponent. Pre prehľadnosť sú jednotlivé komponenty rozdelené do

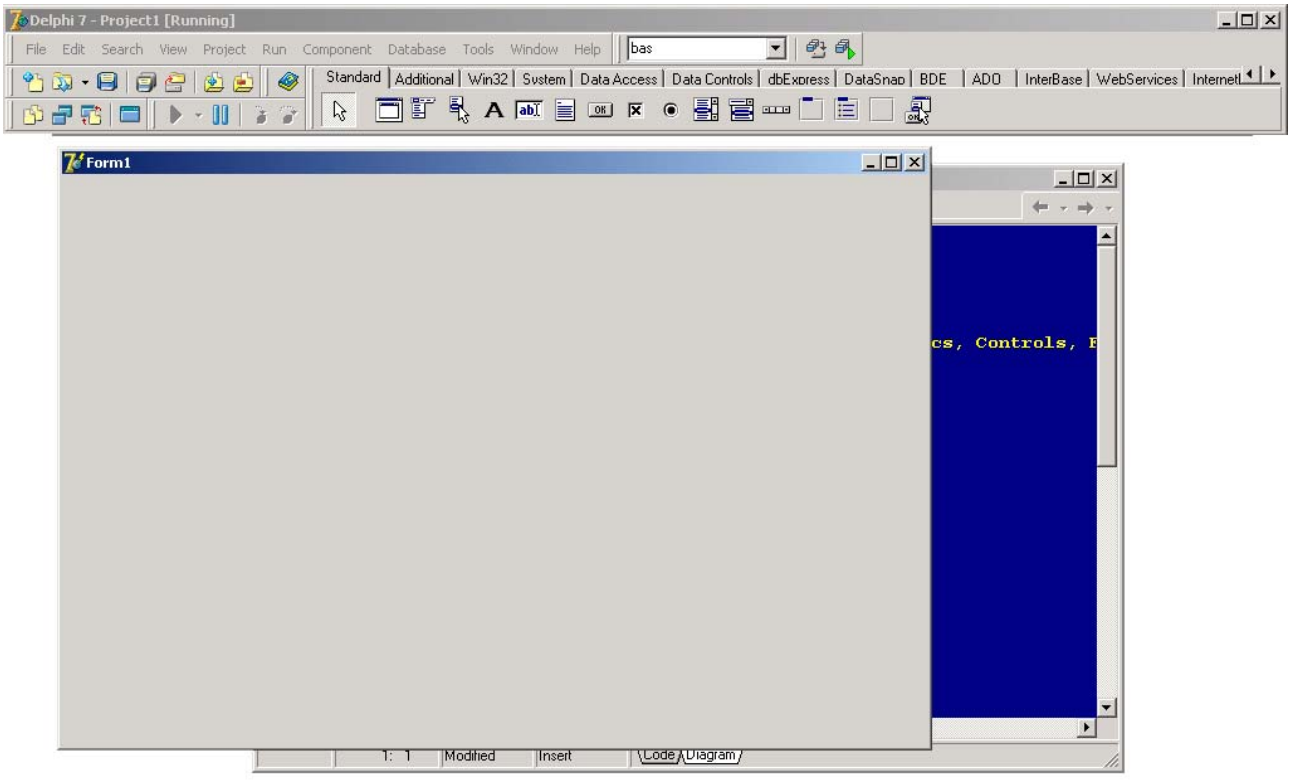
skupín, pričom v začiatkoch určite vystačíme s tými, ktoré sú umiestnené na karte *Standard*.
- *object inspector:* miesto, na ktorom možno nastaviť a meniť vlastnosti aktuálnych komponentov (to sú tie, na ktorých sme nastavení vo formulári). Vyvolať ho možno v ľubovoľnom momente stlačením klávesu **F11**.
- *okno pre zdrojový kód*: je z nášho hľadiska zrejme najzaujímavejšou časťou systému – umožňuje písanie kódu a obsluhovanie udalostí jednotlivých komponentov.

Na obrazovke zvyčajne najviac miesta zaberajú formulár a kód. Často sa prekrývajú alebo niekedy jeden z nich zmizne. Prepínanie, resp. zobrazenie „toho druhého“ dosiahneme vždy klávesom **F12**.

Prvý „program“

Komponenty Delphi sú prispôsobené tak, aby boli schopné vykonávať množstvo činností a meniť množstvo nastavení.

Ak si vezmeme prázdny formulár, ktorý máme k dispozícii okamžite po spustení Delphi (alebo vytvorení novej aplikácie), predstavuje už v tomto momente samostatnú aplikáciu. Bez napísanie jediného riadku kódu dokáže všetko, na čo sme zvyknutí pri oknách Windows. Spustime túto „aplikáciu“ stlačením klávesu **F9** alebo ikonou na riadiacom paneli či položkou menu *Run - Run*.



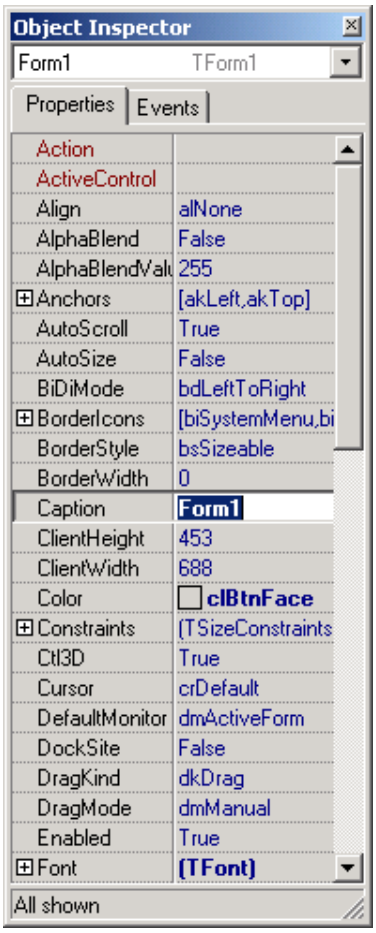
Aktivovaním funkcie odštartujeme kompilovanie zdrojového kódu a v prípade bezchybnosti spustíme aplikáciu. Pre „aplikáciu“, ktorú sme „vytvorili“, sa vytvorí okno s názvom *Form1*. Toto je schopné presunu, minimalizovania, maximalizovania, zmeny veľkosti uchopením za okraj a ťahaním (a to všetko bez jediného riadku kódu).

Kliknutím na „x“ aplikáciu vieme ukončiť.

Na tomto mieste by som rád upozornil na veľmi často opakovanú začiatočnícku chybičku. Je rozdiel, či máte pred sebou okno spustenej aplikácie alebo okno formulára, ktoré možno meniť. Často sa totiž stáva, že študent sa snaží meniť formuláru parametre a diví sa, že tento nereaguje. Ak máte pred sebou formulár s hladkou plochou, ide zrejme o aplikáciu. Ak okno formulára obsahuje bodky uložené do mriežky, ide návrh formulára v prostredí *Delphi*.
Ďalšia vec, ktorá vás informuje o tom, s čím pracujete je panel úloh *Windowsu*. Ak je aktívne tlačidlo *Delphi*, pracujete zrejme s návrhom, ak tlačidlo aplikácie, zrejme ide o spustený formulár.
Niekedy sa stáva, že programátor zmení parametre okna v Delphi, chce si spustením skontrolovať výsledok, ale aplikácia nejde spustiť, tlačidlo spustenia je deaktivované a Delphi na **F9** nereaguje. V takomto prípade pravdepodobne bude aplikácia už spustená v predchádzajúcom stave a treba ju najprv ukončiť. Zasa ju dokážeme identifikovať prostredníctvom panela úloh.

Prišli sme teda k záveru, že na vytvorenie bežiacej aplikácie pod Windows nepotrebujeme prakticky nič. Čo však, ak chceme do nej zakomponovať vlastné požiadavky alebo jej vlastnosti zmeniť?

Parametre komponentu

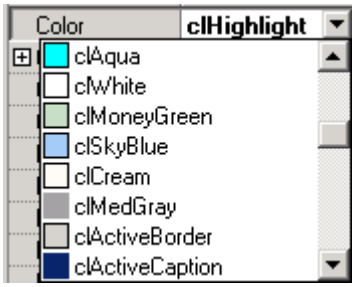


Ako sme už vyššie spomenuli, meniť vlastnosti komponentov nám dovoľuje *Object inspector*. Kliknime si na formulár a vyvolajme jeho vlastnosti prostredníctvom *Object inspectora* (napr. tlačidlom **F11**). Zobrazí sa okno s vlastnosťami uloženými pod seba.

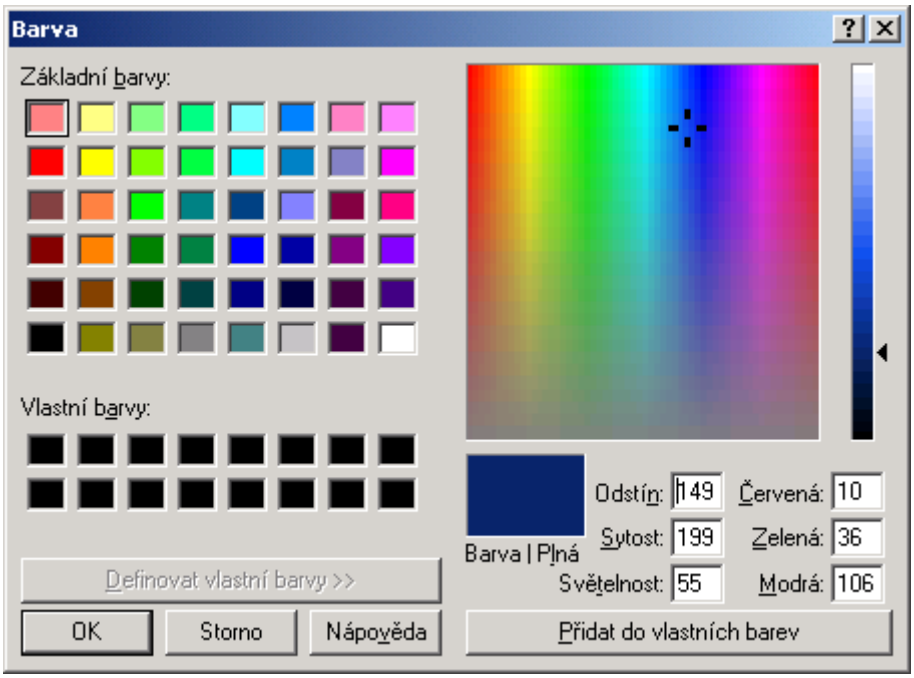
Pokiaľ sa Object inspector nezobrazí, neukončili ste spustenú aplikáciu aktuálneho formulára (pozri vyššie).

Skúsme teraz zmeniť názov (záhlavie) formulára – text, ktorý sa zobrazuje v modrom pruhu aplikácie. Doposiaľ sme v ňom mali zobrazený text „*Form1*“. Vlastnosť, ktorá tento text určuje sa nazýva *Caption* a jej prepísaním sa okamžite zmení i záhlavie formulára.

Podobným spôsobom môžeme zmeniť i farbu formulára. V položke *Color* nájdeme po rozbalení zoznamu preddefinované farby. Tieto môžu byť buď konkrétne (*clWhite*, *clRed*,...) a po našom nastavení nemenné alebo môžu byť prepojené na vlastnosti systému Windows a po zmene jeho farieb cez nastavenia obrazovky sa prispôsobia (*clBtnFace, clHighlight,...*).



Dvojklikom do pravého stĺpca položky vyvoláme systémový color dialóg, v ktorom dokážeme požadovanú farbu „namiešať“.



Zmena farby a záhlavia formulára prostredníctvom *Object inspectora* zrejme nenaplní naše vývojárske očakávania a pravdepodobne budeme požadovať viac (pre začiatok aspoň vypísanie textu, nejaké tie tlačidlá schopné základných činností atď.).

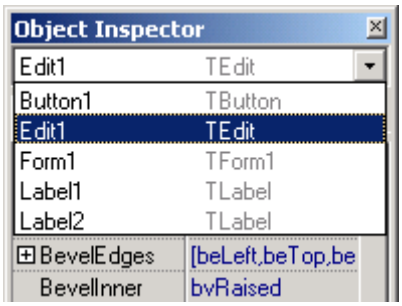
Nevyhnutnosťou bude zoznámenie sa zo základnými komponentami a ich vlastnosťami. Už vieme, že na formulár ich dostaneme kliknutím na komponent z palety a potom kliknutím na miesto formulára, kde má byť komponent umiestnený alebo dvojklikom na komponent (vtedy sa mu umiestnenie určí automaticky).

Label – komponent slúžiaci na zobrazenie textu. Text, ktorý doň vložíme zostane zobrazený po spustení aplikácie. Vlastnosť určujúca text, ktorý chceme zobraziť sa nazýva *Caption* (rovnako ako pri formulári).

Edit – komponent určený na zadávanie textu používateľom a tým pádom umožňujúci komunikovať s aplikáciou. Po spustení aplikácie doň možno vkladať ľubovoľný text, no primárne (okamžite po vložení z panela komponentov i po spustení aplikácie) obsahuje reťazec napr. *Edit1*, ktorý môže pôsobiť dosť rušivo. Vymažeme ho tak, že vyprázdnime položku *Text*, ktorá má rovnaký zmysel ako *Caption* v predošlých prípadoch.

Button – tlačidlo, ktorého zmyslom je zvyčajne po stlačení vykonať nejakú tú operáciu. Dôležitou je pre nás opäť vlastnosť *Caption*, ktorá určuje text zobrazený v tlačidle.

Ak chceme manipulovať s vlastnosťami konkrétneho komponentu, je ho potrebné obvykle najprv označiť (napr. kliknutím myši). O tom, že s ním skutočne manipulujeme, nás informuje aj prvý riadok *Object inspectora*. Pomocou neho sa môžeme medzi jednotlivými komponentami umiestnenými na formulári aj prepínať (ak nevieme nájsť komponent na formulári, alebo je ich množstvo už značne neprehľadné).
Pokiaľ sa parametre nemenia podľa vášho očakávania, skontrolujte si, či naozaj ide o vybraný komponent.



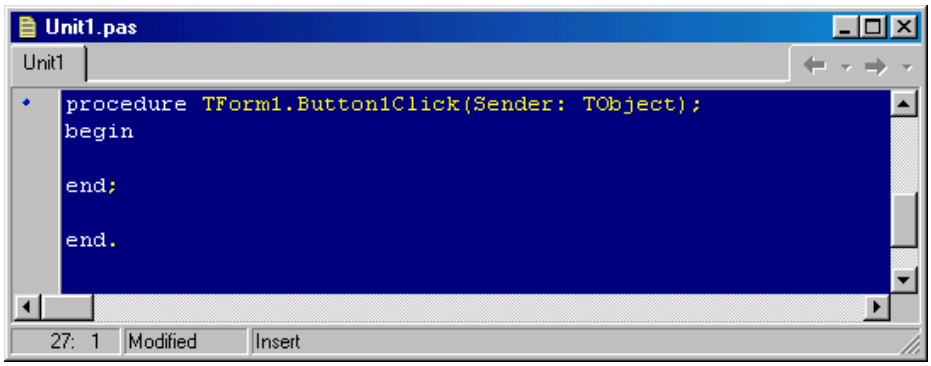
S touto trojicou (štvoricou ak rátame aj formulár) komponentov pre začiatok vystačíme.

Prvá ozajstná aplikácia

Vytvorme na začiatok veľmi jednoduchú aplikáciu, ktorá bude schopná po stlačení tlačidla zmeniť farbu formulára – postačia tri: červená, zelená a modrá).
Na formulár vložte trojicu tlačidiel, napíšte do nich názvy príslušných farieb a...
... konečne môžeme začať programovať.

Ako vysvetliť systému, že po kliknutí na tlačidlo by bolo dobré niečo vykonať. Cesty sú dve:

- prvá rýchlejšia a jednoduchšia avšak nie univerzálna nás vedie k tomu, aby sme na tlačidlo dva razy klikli (pozor, klikajte naozaj na tlačidlo, nie na formulár). Dvojklik nás prenesie do okna zdrojového kódu a do procedúry, ktorá by mala vyzerať podobne ako na obrázku.



Názov procedúry hovorí, že ide o tlačidlo *Button1* a udalosť *Click*, t.j. kód, ktorý sem napíšeme sa vykoná pri kliknutí na toto tlačidlo.

*Ak sa vytvorila procedúra s iným názvom (nie button a nie click), vráťte sa klávesom **F12** do okna formulára a skúste kliknúť znova.*

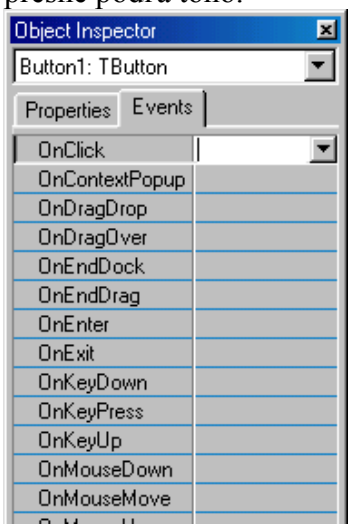
Medzi *begin* a *end* môžeme teraz vložiť príkazy, ktoré zabezpečia zmenu farby formulára na červenú:

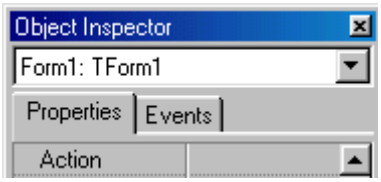
Form1.Color:=clRed;

Formulár sa volá *Form1*. Tento jeho názov môžeme overiť vo vlastnostiach v položke *Name* (pozor *Name* nie je to isté, čo *Caption*).
Chceme mu zmeniť vlastnosť *Color* - vlastnosť sa od komponentu oddeľuje bodkou.
Samotné priradenie realizujeme prostredníctvom *:=*.

Priraďovanú farbu môžeme zadať číslom (pokiaľ si pamätáme) alebo konštantou, ktorú systém pozná: *cl* znamená, že ide o farbu (color), *Red*, že farba bude červená.
Takýmto spôsobom fungujú konštanty v mnohým prípadoch, pre nás postačí zatiaľ vedieť, že druhé tlačidlo potrebuje *clBlue* a tretie *clGreen*.

- druhá možnosť je možno trochu zložitejšia, avšak poodhalí nám princíp toho, čomu sa hovorí **udalosťami riadené programovanie**.
  Windows pracuje tak, že každé stlačenie klávesu, kliknutie alebo pohyb myši vyvolá udalosť. Túto udalosť potom môžeme ošetriť. Pokiaľ sa k nej nevyjadrujeme (nenapíšeme žiaden kód), nedeje sa nič, ak čo-to naprogramujeme, postupuje sa pri vzniku udalosti presne podľa toho.



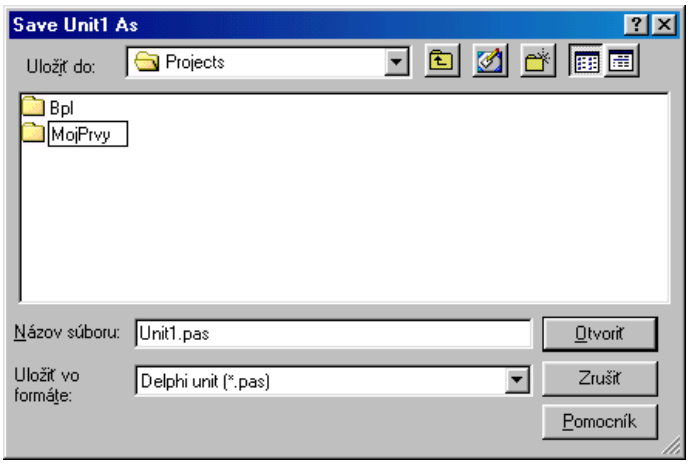


Kliknime si na ľubovoľné tlačidlo a vyvolajme *Object inspectora*. Možno ste si všimli, možno nie, *Object inspector* pozostáva z dvoch záložiek – *properties* (vlastnosti) a *events* (udalosti).
O *propertiesoch* už vieme, že nastavujú vlastnosti (vzhľad komponentu), *events* obsahuje udalosti, na ktoré je schopný komponent reagovať. Pre *Button* sú k dispozícii tie, ktoré vidíte na obrázku
Nás zrejme zaujme prvá udalosť – *onClick*. Popisuje to, čo sa má stať pri kliknutí na objekt (tlačidlo). Ak do položky (prázdneho poľa za *OnClick*) dvojklikneme, dostaneme sa na to isté miesto ako v predchádzajúcom prípade po dvojkliku na tlačidlo.

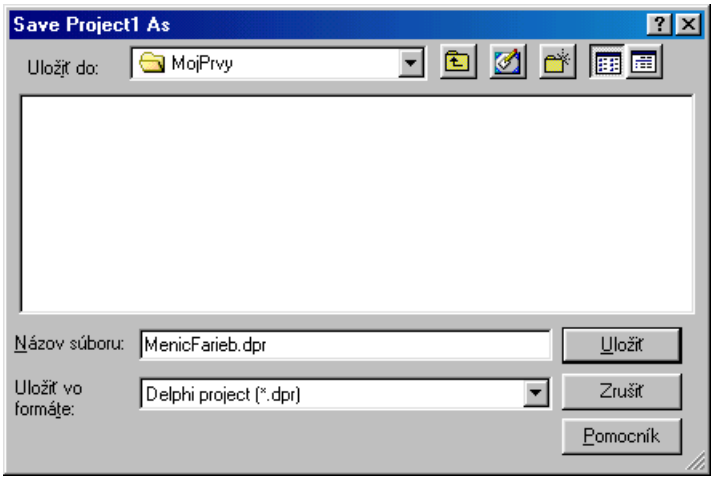
Mohli by ste namietnuť, že prvý prípad je oveľa jednoduchší a rýchlejší, s čím možno len súhlasiť. Jeho nevýhodou je však to, že vždy vyvoláva len jednu udalosť (každý dvojklik na tlačidlo vždy vyvolá udalosť *onClick*). My však niekedy potrebujeme ošetriť aj iné udalosti. Napr by sme chceli, aby sa niečo udialo, keď budeme pohybovať myšou nad tlačidlom (udalosť *onMouseMove*), alebo ak stlačíme tlačidlo *Delete* (udalosť *onKeyPress*) atď.
K týmto udalostiam máme prístup len cez záložku *Events*

Zloženie aplikácie v Delphi
Uložme teraz našu aplikáciu a pokúsme sa ju preniesť na iný počítač tak, aby sme s ňou mohli ďalej pracovať.

Na uloženie všetkých súčastí naraz slúži ikona  , ktorá uloží všetko neuložené. Pokiaľ budete ukladať aplikáciu, postupnosť bude zrejme nasledovná:



Ako prvá časť sa ukladá *unit* – ide o obsah formulára, komponentov na ňom a zdrojového kódu. Pre každú aplikáciu odporúčame vytvoriť osobitný priečinok a *unitom* dávať také mená, aby ste vedeli čo obsahujú.



Za unitmi nasleduje uloženie projektového súboru – tento obsahuje zoznam všetkých použitých unitov, inicializáciu, štart a prípadné ukončenie aplikácie.

Bez projektu unit nespustíte!!!!!!!!!!!!!

Tento názov sa bude zobrazovať jednak ako názov aplikácie jednak pri vytvorenej ikone, a jednak na paneli úloh po spustení aplikácie.

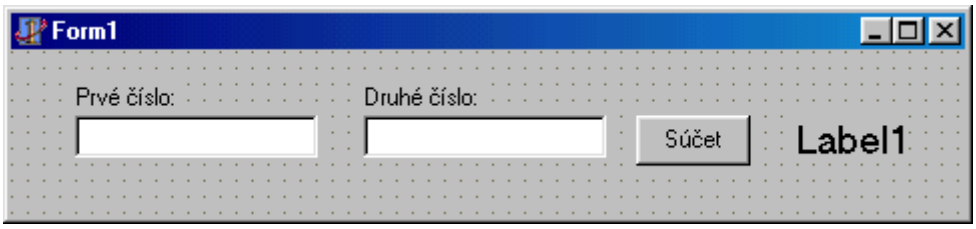
Po úspešnom uložení sa môžeme pozrieť na obsah priečinka, kam sme výsledok uložili.

Názov	Veľk...	Typ
MenicFarieb.cfg	1 kB	Súbor CFG
MenicFarieb.dof	2 kB	Súbor DOF
MenicFarieb.dpr	1 kB	Delphi Project
MenicFarieb.res	1 kB	Súbor RES
Unit1.dfm	1 kB	Delphi Form
Unit1.pas	1 kB	Delphi Source File
MenicFarieb.exe	291 kB	Aplikácia
Unit1.dcu	3 kB	Súbor DCU

Z tohto zoznamu sú dôležité nasledovné súbory:
dpr – ide o súbor projektu,
pas – zdrojový kód unitu napísaný užívateľom,
dfm – zloženie (vzhľad a zoznam komponentov) formulára pridružené k zdrojovému kódu s rovnakým názvom (vytvára sa automaticky pri ukladaní zdrojového kódu).

Bez týchto súborov aplikáciu nespustíte. Ostatné obsahujú nastavenia projektu (majú názov totožný s projektom, odlišujú sa len koncovkou) a skompilované unity (*dcu*).

Kalkulačka
Kalkulačka patrí k jednoduchým programom, na ktorých sa dajú vysvetliť a pochopiť základné vlastnosti aplikácií v Delphi.
Vytvorme teda najprv aplikáciu, ktorá bude sčítavať dve celé čísla.



Ako prvé potrebujeme zrejme vytvoriť prostredie. Mohlo by byť podobné tomu na obrázku.. Vložené texty sú objekty *Label*, ktorým sme zmenili vlastnosť *caption* – napísali sme do nej text, ktorý chceme mať zobrazený.
Textové polia na zadávanie čísel sú objekty *Edit*, ktoré síce po vložení obsahujú vísaný text (*Edit1, Edit2...*), ale ten sme zrušili jeho vymazaním vo vlastnosti *Text*. Tlačidlo je *Button* so zmenenou vlastnosťou *Caption*, no a na zobrazenie výsledku použijeme opäť *Label*. Jeho obsah môžeme cez *Caption* prepísať na *0*, alebo pokojne nechať taký ako je.

Všimnime si jednu vec: text, ktorý sa má zobrazovať v objekte sa zvyčajne nastavuje pomocou vlastnosti *Caption* (*Label*, *Button*, *Form*,....), prípadne v niekoľko málo prípadoch vo vlastnosti *Tetx* (*Edit*).

Po vytvorení prostredia nám zostáva už napísať "len" obslužný kód pre vykonanie operácie.
Prvá otázka: kedy sa má sčítavanie spustiť?
Prvá odpoveď: zrejme keď stlačíme tlačidlo.
Druhá otázka: ako povieme, že kód sa má vykonať až vtedy, keď sa tlačidlo stlačí?
Druhá odpoveď: zapísaním kódu do udalosti pri kliknutí (*onClick*). Nastaviť, aby sa kód naozaj vykonal práve vtedy možno dvoma spôsobmi (uviedli sme ich už aj vyššie): dvojklikom na tlačidlo (Delphi nás prepne do kódu, kde sa automaticky vytvorí procedúra s názvom *Button1Click*) alebo prepnutím sa v *Object inspectore* na záložku udalostí (*Events*) a dvojklikom do položky *OnClick* (vytvorí sa presne tá istá procedúra).
V oboch prípadoch je teda výsledok rovnaký...
Tretia a posledná otázka: ako napísať kód na vykonanie sčítavania?
Tretia a posledná odpoveď: veľmi jednoducho – pomocou jazyka Pascal a vlastností vložených komponentov.
Samozrejme, táto odpoveď potrebuje trochu presnejšie vysvetlenie. V prvom rade si treba uvedomiť, že v Delphi naozaj nejde o nič iné ako o kombináciu štandardného Pascalu (so všetkými jeho kladmi i zápormi) a vytvorených objektov – komponentov. Každý z komponentov má kdesi skrytý svoj zdrojový kód a ten je zvyčajne äaspoň pre nami zatiaľ používané komponenty) tiežnapísaný v Pascale.
Pri programovaní teda používame zápis deklarácie, príkazov i procedúr rovnako ako v Pascale.
No a o tom budú aj nasledovné riadky.
Potrebujeme sčítať dve celé čísla. Prvým krokom by teda mohla byť ich deklarácia v tele automaticky vytvorenej procedúry:

```
procedure Tform1.Button1Click(Sender:TObject)
var a,b,vysledok:integer

begin
```

```
end;
```

Do premennej *a* načítame prvé číslo, do premennej *b* druhé. *Vysledok* bude obsahovať ich súčet. Pred samotným odštartovaním je však potrebné zdôrazniť, že akýkoľvek reťazec (je jedno či text alebo číslo), ktorý sa zapisuje alebo číta z *Editu* je automaticky a nevyhnutne typu *string*. Z tohto dôvodu potrebujeme pred uložením zadanej hodnoty vykonať prevod zo *stringu* na celé číslo. Používa sa na to funkcia *StrToInt* (string na integer). Zápis potom bude vyzerať nasledovne:

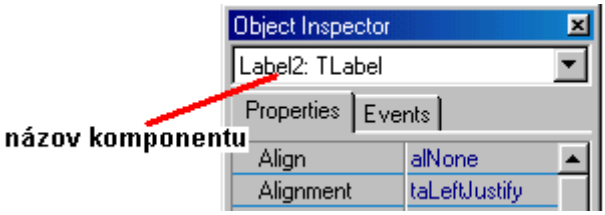
```
procedure Tform1.Button1Click(Sender:TObject)
var a,b,vysledok:integer

begin
 a:=StrToInt(Edit1.Text);
 b:=StrToInt(Edit2.Text);

end;
```

Do premennej *a* vložíme číslo, ktoré sa nachádza v *Edit1*. Vlastnosť, ktorá nás informuje o tom, čo je v *Edite* zapísané je *Text*. Prečítaný obsah je typy *string*, pre ktorý (ako si iste pamätáme z Pascalu) sčitovanie nefunguje. pomocou funkcie *StrToInt* ho prevedieme na číslo a uložíme do premennej *a*. Podobne druhé číslo prejde do premennej *b*.

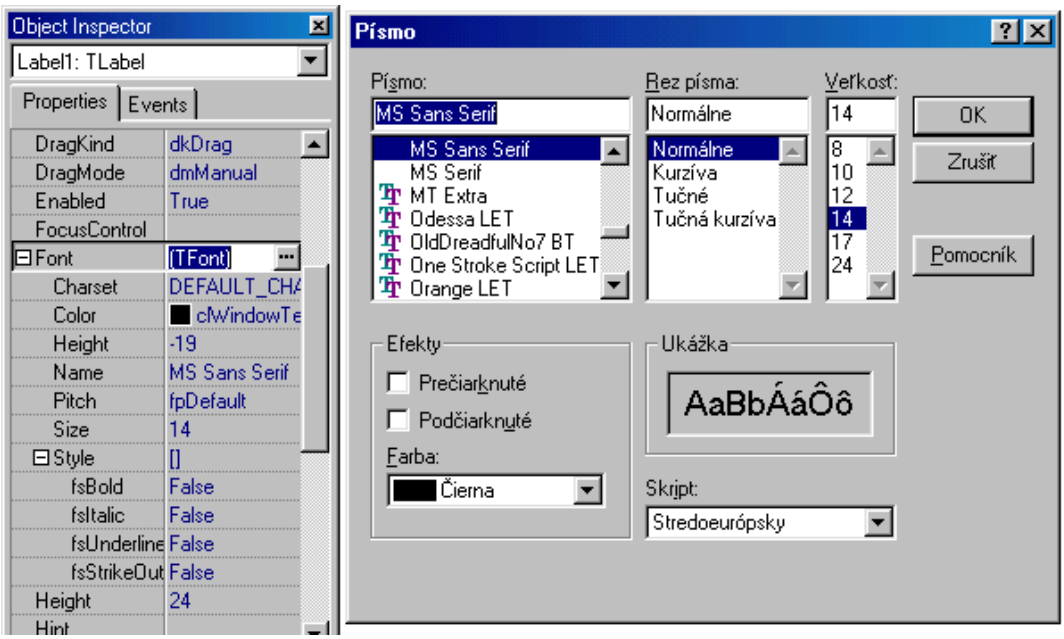
```
procedure Tform1.Button1Click(Sender:TObject)
var a,b,vysledok:integer

begin
 a:=StrToInt(Edit1.Text);
 b:=StrToInt(Edit2.Text);
 vysledok:=a+b;
 Label1.Caption:=IntToStr(vysledok);
end;
```

Súčet vykonáva klasické priradenie a výsledok sa zobrazí v *Labeli*. Zobrazovaný text *Labelu* je uložený vo vlastnosti *Caption*, ktorá je opäť typu string. Číslo na string prevedieme pomocou obrátenej funkcie – *IntToStr*.
A výsledok je na svete...

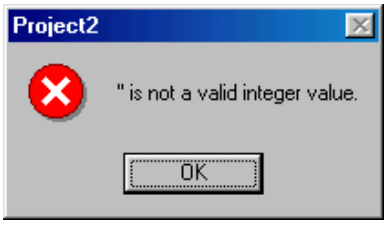
V prípade, že váš *Label* sa nevolá *Label1*, treba jeho názov upraviť podľa svojho. Názov nájdete vo vlastnosti *Name* alebo rýchlejšie v hornej časti *Object inspectora* po kliknutí na príslušný objekt.



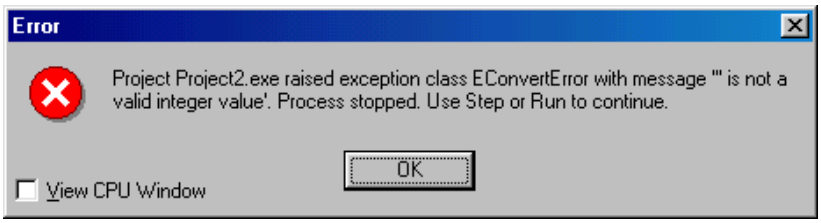
Možno ste si všimli, že *Label* s výsledkom je na našom obrázku podstatne väčší ako *labely* s popisným textom k *Editom*. Je to spôsobené zväčšením písma – zmenou fontu. Prakticky každý objekt obsahujúci text má k dispozícii vlastnosť *Font*, ktorá po dvojkliku na pravej strane zobrazí štandardný *FontDialog* alebo po rozbalení dovoľuje meniť vlastnosti písma po položkách.



Na záver by to chcelo upozorniť, že funkcia *StrToInt* je pomerne hlúpa – ak sa pokúsite previesť na číslo prázdny text alebo neplatné číslo (písmená, medzery...) – vyhodí chybu a preruší vykonávanie programu. Nedivte sa preto, ak sa v prípade nesprávneho čísla zobrazí hláška:



alebo



Informuje len o tom, že zadaný reťazec nie je celé číslo. Vyriešiť tento problém možno napr.pomocou procedúry *Val* známej už z Pascalu.

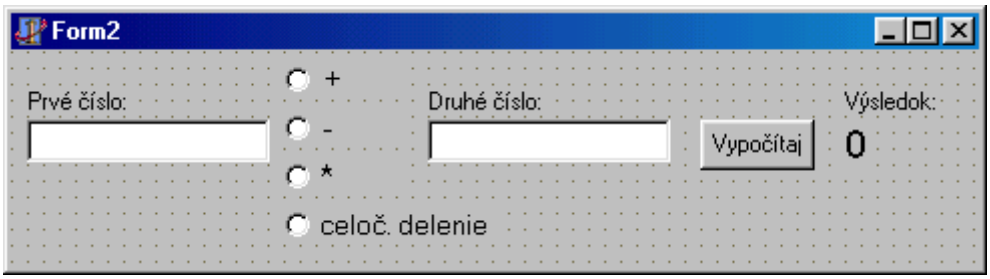
Vytvorte ďalšie tri tlačidlá pre súčin, rozdiel a podiel (môže byť celočíselné delenie pomocou div a mod) alebo desatinné (v takom prípade bude výsledkom reálne číslo, na čo treba myslieť pri deklarácii i pri prevode na string – používa sa funkcia FloatToStr).

Postup je analogický ako v predchádzajúcom prípade, mení sa len znamienko pri výpočte.

RadioButton

Upravte kalkulačku tak, aby výpočet spúšťalo jediné tlačidlo, a aby sa operácia dala vykonať podľa nastavenia v RadioButtone.

Formulár môže vyzerať napr. nasledovne:



Prázdne krúžky sú *RadioButtony*, ktoré majú tú vlastnosť, že len jeden z nich môže byť zapnutý (bude obsahovať čierny kruh). Momentálne nie je zapnutý ani jeden, ak však chceme nastaviť niektorú z operácií ako primárnu (automaticky sa zaškrtne po spustení), môžeme tak urobiť pomocou vlastnosti *Checked*. V prípade, že je táto nastavená na *False*, *RadioButton* zaškrtnutý nie je, ak sa nastaví na *True*, zaškrtne sa. Nastavenie (zaškrtnutie) ktoréhokoľvek *RadioButtonu* na formulári má za následok vypnutie všetkých ostatných (naraz môže byť vybraný len jeden).

Pokiaľ by ste potrebovali mať vybraných naraz viac položiek, môžete použiť *CheckBox*, *RadioGroup* alebo *Panel*.

Nastavme teraz ako primárne zvolenú operáciu súčtu a pustime sa do písania kódu. Jediná udalosť, ktorú potrebujeme obslúžiť je opäť kliknutie na tlačidlo.
V prvom rade prečítame čísla zadané v *Editoch*, potom zistíme, ktorý *RadioButton* je vybraný, podľa neho vykonáme výpočet a napokon výsledok vložíme do *Labelu*.

```
procedure Tform1.Button1Click(Sender:TObject)
var a,b,vysledok,zvysok:integer


begin
 a:=StrToInt(Edit1.Text);
 b:=StrToInt(Edit2.Text);
 if RadioButton1.Checked then vysledok:=a+b;
 if RadioButton2.Checked then vysledok:=a-b;
 if RadioButton3.Checked then vysledok:=a*b;
 if RadioButton4.Checked then begin
  vysledok:=a div b;
  zvysok:=a mod b;
 end;

 Label1.Caption:=IntToStr(vysledok);
 if RadioButton4.Checked then
   Label1.Caption:= Label1.Caption + ', zvysok: ' + IntToStr(zvysok);
end;
```

Pre prvé tri *radiobuttony* by malo byť všetko jasné – ak sú zaškrtnuté (čo vyjadruje ich vlastnosť *checked* nastavená na *true*), tak výsledok vypočítame príslušným spôsobom. Ak sme zvolili celočíselné delenie, potrebujeme okrem výsledku zistiť aj zvyšok.
Pre výpis v prvých troch prípadoch stačí priradiť výsledok do *Labelu*. Pokiaľ sme celočíselne delili, potrebujeme vypísať aj zvyšok. K výsledku už zobrazenému v *Labeli* pridáme ďalší

textový reťazec: *zvysok:* a prevedenie jeho hodnoty na *string*. Výsledný reťazec potom môže vyzerať napr. takto:

```
2, zvysok: 5
```

Všetko je na prvý pohľad v poriadku, ak však pri delení zadáme ako deliteľa nulu, aplikácia nám vyhodí chybu. Na ošetrenie by postačila podmienka a v prípade nuly výpis upozornenia do *labelu*. Ak však chceme dodržať štandardy, patrí sa v takomto prípade vypísať upozornenie do osobitného okna.
Jedným z najjednoduchších spôsobov je použitie funkcie *ShowMessage*, ktorá zobrazí v osobitnom okne text. Kód môže vyzerať takto:

```
...
if RadioButton4.Checked then begin
  if b=0 then begin
   ShowMessage('Nulou deliť nemožno');
   exit;
  end
  vysledok:=a div b;
  zvysok:=a mod b;
 end;
...
```

V prípade zadania nuly teda zobrazíme správu a príkazom *exit* opustíme procedúru – nevykonajú sa teda žiadne operácie nasledujúce za týmto príkazom (výpočet, výpis).
Správe bude vyzerať podobne ako táto: